УДК 681.3.07
ББК 32.973.26–018.2.75
А19

А19 Под редакцией Тани Х. Шорт и Тарна Адамса
Процедурная генерация в гейм-дизайне / пер. с англ. М. С. Рыжиковой. – М.: ДМК Пресс, 2020. – 344 с.: ил.

ISBN 978-5-97060-860-9

Книга представляет собой сборник статей, посвященных приемам процедурного повествования в играх и не только.

Процедурная генерация – не новое, но пока еще малоизученное явление, и каждый проект преподносит свои вызовы. Подчас разработчикам сложно оценить, какую часть игры лучше написать вручную, а какую доверить процедурному генератору. Авторы материалов, вошедших в сборник, расскажут о том, как оформить сложное нелинейное повествование в игре, как продумать интересных персонажей, диалоги и описания, которые будут генерироваться произвольно или в зависимости от действий игрока, и как средствами компьютерных технологий донести до игроков непростые философские и этические идеи.

Помимо игр методы процедурной генерации рассматриваются на примере настройки твиттер-ботов, гадания на картах таро и даже некомпьютерных видов активности наподобие интерактивных прогулок.

Издание адресовано разработчикам компьютерных игр и приложений, предназначенных для широкого круга пользователей.

УДК 681.3.07
ББК 32.973.26–018.2.75

ISBN 978-1-49195-446-1 (англ.)
ISBN 978-5-97060-860-9 (рус.)

Оглавление

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и CRC Press очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Вступление

Надеюсь, Таня понимала, на что идет, когда попросила меня написать предисловие. Я в основном известен по очень, очень специфическим ролевым играм в стиле «выбери свое собственное приключение» (*Planescape: Torment*, *Fallout 2*, *Fallout: New Vegas*, *Knight of the Old Republic II* и многим другим). В результате моей первой – и неправильной – реакцией было желание написать что-то очень оригинальное и мудреное по поводу этой книги. Но, так как я не слишком умен, идея была обречена на провал – к великому счастью.

Например, я рассматривал идею о написании этого предисловия как процедурно сгенерированный текст. Но потом я решил, что Таня меня убьет, и было бы разумно предпринять нечто более практичное… Особенно я укрепился в этой мысли после прочтения некоторых глав книги, которые очень полезны. И я чувствовал бы себя виноватым, если бы не сказал ничего по существу.

Итак – имейте это в виду – у меня для вас есть три крошечные (пусть и не высшего качества) «жемчужины мудрости», которые позволят вам взять быстрый старт при погружении в материал книги. А именно:

1) мой ограниченный опыт работы с процедурным контентом в большинстве ролевых игр, с которыми я имел дело, и понимание того, почему это может быть проблемой;
2) несколько соображений по поводу того, что «случайный» не всегда означает случайный, и…
3) и, наконец, опасность создания специального контента для в основном процедурной игры. Это своего рода оппозиция первому пункту, но таким образом круг изящно замыкается.

А теперь ближе к делу!

Итак, RPG на протяжении почти всей моей карьеры характеризовались весьма специфическим контентом[1].

Их цель – давать как можно более удачный отклик на действия игрока. Даже когда вы знаете содержание и последствия действия, получить специфическую реакцию – все еще непростая задача. Иногда ответные реплики NPC («вы слышали, что случилось на шахте, по словам шахтеров?») могут не совпасть с опытом игрока на все сто («я уверен, что всех там переубивал, и никто не ушел живым»).

---

[1] Контент – информативное содержание игры: графическое, аудиальное, текстовое. Это совокупность всего того, что может найти игрок в конкретной сцене: вещи и их названия, персонажи и их имена – *Прим. перев.*

Кроме того, как понимают многие издатели – и оттого предпочитают уходить от темы, – этот специфический контент слишком дорого обойдется, если реализовать его правильно. Таким образом, некоторый уровень процедурного содержания (случайные встречи – ну, те, которые действительно случайны) всегда был чем-то, о чем стоило заботиться и что следовало предлагать там, где только можно, поэтому не каждый момент игры должен был быть написан вручную. Вам нужна какая-то случайность, – немного здесь, немного там… теперь вы понимаете. Некоторое случайное содержание и случайные встречи, чтобы конкретизировать специфические байты.

Но… затем возникает другая проблема, которая подводит меня ко второму важному пункту. А что такое *случайность*? Обязательно задавайте этот вопрос! Определение чего-либо «случайного» в системе будет сильно зависеть от того, кого вы спрашиваете.

Обычно «случайный» контент – это действительно случайный набор вызовов контента независимо от того, используете ли вы его для имен персонажей, для фона, для квестов, для оружия… для чего угодно. Чем более абстрактно содержание, тем меньше вам придется беспокоиться об истинной случайности, но тем более специфичны названия (персонажи, фоны) и тем значимым оказывается представление разработчиков о «рандоме» и «псевдорандоме». Иногда разработчики (и игроки) хотят что-то действительно случайное, иногда – что-то более похожее на «перетасовку» контента, подобное перемешиванию треков в списке воспроизведения. Это может быть приравнено к другой технике, которую я люблю называть «псевдослучайной».

Концепция «псевдослучайности», о которой позволяет судить история одного разработчика, рассказанной мне Тимом Кейном (Tim Cain) (*Fallout*, *Arcanum* и др.), имеет несколько дополнительных параметров, в отличие от простой концепции случайности. «Псевдослучайность» – это на самом деле придуманный мной термин, и я не думаю, чтобы Тим использовал такое название. Вкратце суть «псевдослучайности» сводится к ситуации, когда разработчик или игрок *утверждает*, что ему нужен генератор случайных чисел, а на самом деле хочет добиться *иллюзии* случайности.

Это происходит потому, что в истинной системе генерации случайных чисел возможна ситуация, когда два числа или экземпляра контента повторяются подряд. Однако редко кто из разработчиков игр хочет этого. Игроки тоже не захотят этого, поскольку это упрощает и обедняет игровой мир. Даже если числа и экземпляры контента не повторяются подряд, может случиться так, что истинная система случайной генерации вызовет повторение двух чисел, прежде чем полный список чисел/экземпляров контента будет показан игроку. Гейм-дизайнеру это не по нраву. Да и игроки не будут в восторге.

Псевдорандомисты предпочитают, чтобы цикл генератора случайных чисел проходил все возможные перестановки один раз, не повторяясь, пока все числа не будут исчерпаны, а затем повторять процесс при условии, что число, сгенерированное по завершении последовательности, не является последним числом предыдущей последовательности.

Ну и ну! И да, это считается «случайным». Поэтому обязательно спросите, какой случайности ожидают разработчик и игрок.

В целом иногда столь же запутанные, как их типы, случайности могут обогатить сценарии «особых случаев» – но (см. выше пункт 3) не придавая им чересчур большого значения в процедурной игре. На самом деле включение специального содержания случайностей в процедурную (в значительной части) игру может быть довольно опасным.

Чтобы пояснить свою позицию, скажу, что в своей работе с процедурными играми я в основном опирался на знания, которые мне передали Джастин Ма (Justin Ma) и Мэтью Дэвис (Matthew Davis), руководители игровых проектов. Я имел удовольствие работать с ними над *FTL: Advanced Edition* и *Into the Breach*. У меня также была возможность поработать над другими процедурными проектами (*Overfall*). И это чрезвычайно перспективное направление – я работал над рядом проектов, где создание процедурного контента становится все более и более важным, и способность правильно получить такой контент еще более важна. Поэтому рассматривайте эту книгу как мощное учебное пособие, которое поможет вам найти хорошую работу, или как справочный материал, если вы уже вовлечены в создание процедурного контента.

В любом случае одним из ключевых уроков в процедурных играх, над которыми я работал, был следующий: будьте осторожны с контентом для особых случаев. Подобные находки в процедурной игре (обычно это уникальная встреча) могут показаться отличной наградой для игрока, но те же события начинают утомлять вас, если вы сталкиваетесь с ними и на втором прохождении, и на третьем… и на четвертом… А в процедурной игре это происходит чаще, чем вы думаете. Игроки быстро раздражаются и приходят в уныние, очередной раз проходя детально проработанный этап, который вы всунули в игру. Это гораздо скучнее обычных модульных встреч, к которым они привыкли во время игры.

Итак, есть один метод, важность которого подчеркивается процедурным содержанием: «не выделяться слишком сильно»; это может относиться к конкретному случайному событию, к специальному NPC или даже к необычному, выделяющемуся на общем фоне озвучиванию (или стилистике субтитров). Мы с этим столкнулись, когда прорабатывали реплики персонажей RPG, такие как «*Я угораю*», «*Я тебя упустил*» или «*О нет, Таня хочет, чтобы я побыстрее закончил это предисловие!*» Когда вы слышите или читаете такую фразу, вы сразу же чувствуете, что она выбивается из

общего контекста, так что придется вам наступить на горло собственной песне и поменьше оригинальничать.

Это означает, что вы должны в некотором смысле понизить насыщенность создаваемого вами процедурного контента. Это весьма неожиданный метод, который стоит принять на вооружение гейм-дизайнеру, но, когда вы поиграете в игру достаточно долго, вы это оцените. Когда вы услышите ТАКИЕ ВОТ ЯРКИЕ СЛОВА более пяти раз, вы уже будете на взводе, потому что они И ТАК КРЕПКО ЗАСЕЛИ у вас в голове! Выделяющиеся фрагменты на самом деле показывают слабость всей вашей процедурной системы. Создается впечатление, что одна строчка текста пытается затмить все остальные, которые просто делают свою работу.

Вот и все мои «жемчужины мудрости». Эта книга содержит гораздо больше идей – притом гораздо лучших. Я надеюсь, что она окажется для вас ценной. Процедурный контент – малоисследованная тема в игровой индустрии, и реализовать его подчас сложно, но эта книга может помочь и указать вам путь.

Крис Авеллон (Chris Avellone),
независимый рецензент

О редакторах

Таня Шорт (Tanya X. Short) – капитан Kitfox Games, независимой студии, разрабатывавшей *Boyfriend Dungeon*, *The Shrouded Isle*, *Moon Hunters* и другие игры. Таня – соучредитель и один из директоров Pixelles, интерсекционально-феминистской некоммерческой организации по разработке игр. Хозяйка двух кошек, которые не питают ни малейшего интереса к игровому контенту – ни к процедурному, ни к какому бы то ни было.

Тарн Адамс (Tarn Adams) – соучредитель компании Bay 12 Games наряду с его братом Заком. Вместе они работают над фэнтези-симулятором *Dwarf Fortress* – одной из первых видеоигр, приобретенных Музеем современного искусства в Нью-Йорке. Тарн создает и отлаживет процедурные интерактивные повествовательные проекты на протяжении более 20 лет.

Составители

Тарн Адамс (Tarn Adams), Bay 12 Games
Мэтью Бозарт (Matthew Bozarth), Undead Labs
Джордж Бакенгэм (George Buckenham)
Джеффри Кард (Geoffrey Card), Undead Labs
Д-р Кейт Комптон (Dr. Kate Compton)
Дэниэл Кук (Daniel Cook), Spry Fox
Д-р Майкл Кук (Dr. Michael Cook), Queen Mary University of London
Бруно Диас (Bruno Dias)
Риад Джемили (Riad Djemili), Maschinen-Mensch
Алекс Эпштейн (Alex Epstein), Compulsion Games
Марта Фиджек (Marta Fijak), 11 Bit Studios
Джейсон Гринблат (Jason Grinblat), Freehold Games
Бенджамин Хилл (Benjamin Hill)
Юрие Хорнеман (Jurie Horneman)
Джон Ингольд (Jon Ingold), Inkle
Дариус Каземи (Darius Kazemi)
Тим Кинан (Tim Keenan)
Йонгву Ким (Jongwoo Kim), Kitfox Games
Дэниэл Клайн (Daniel Kline)
М. Лейзер-Уолкер (Mx. Lazer-Walker)
Стивен Лампкин (Steven Lumpkin), Guerrilla Games
Джимми Мейер (Jimmy Maher), The Digital Antiquarian Blog
Кэт Мэннинг (Cat Manning)
Джилл Мюррей (Jill Murray), Discoglobe Interactive
Ричард Раус III (Richard Rouse III), Paranoid Productions
Элан Раскин (Elan Ruskin)
Адам Зальтцман (Adam Saltsman), Finji
Эмили Шорт (Emily Short)
Таня Шорт (Tanya X. Short), Kitfox Games
Кевин Сноу (Kevin Snow)
Якоб Стокальски (Jakub Stokalski), 11 Bit Studios
Йорген Тьерно (Jørgen Tjernø), Undead Labs

Часть I

Введение

Некоторые антропологи считают, что первые интерактивные истории были изобретены в отблесках костра, когда старейшины племени в своих выступлениях адресовали отдельные реплики аудитории и рассчитывали на ее реакцию. Если это так, то идея индивидуального авторства уже вторична. А видеоигры – еще более современное изобретение, и применительно к ним системное повествование исследуется крайне редко.

Индустрия видеоигр сейчас – большая и быстро растущая область, в ней задействованы десятки тысяч работников по всему миру, но даже при всей нашей специализации очень немногие возьмут на себя смелость называться экспертами в процедурном повествовании. Чтобы подготовить этот сборник, редакторам пришлось пообещать авторам отдельных статей, что их мастерство не будет преувеличиваться. Процедурный нарратив – явление не новое, но это та сфера, в которой еще остается масса сомнений и оговорок..

Авторы статей, включенных в этот сборник, будут использовать термины «процедурные» и «истории» в несколько разном значении, опираясь на свой личный опыт.

Тем не менее редакция надеется, что вы найдете:

- новые идеи, подходы и философские концепции для рассмотрения;
- новые примеры разработок и процессов, реализованных на практике;
- инструменты и ресурсы, которые можно использовать в ваших собственных проектах.

В этой вводной части мы попытались представить общий, широкий взгляд на проблему – поговорить о важности и содержательности игрового повествования, управляемого автоматически. Книга представляет собой подборку статей, и в каком бы порядке вы бы ни взялись их читать, мы надеемся, что вы с интересом отнесетесь к подходам каждого из наших авторов, приветствуя бесконечные возможности профессионального роста в этой богатой и сложной области.

Глава 1

Начало работы с генераторами

Д-р Кейт Комптон

Выдержка из статьи «Итак, вы хотите построить генератор?»
galaxykate0.tumblr.com

Учитывая, что в мире существует столько всевозможных видов генераторов, что здесь можно посоветовать? Это будет зависеть от того, какой генератор вы хотите построить. Так что я задам вам ряд наводящих вопросов, а затем уж дам совет – смотря по тому, что вам нужно.

Первый вопрос: *что вы собираетесь делать?*

Запишите то, что будет делать ваш генератор. Назовем это «артефактом», но это может быть что угодно: процедурные птицы, генерируемые истории, анимированная хореография, рецепты гаспачо, квесты в RPG, шахматные партии.

Теперь самое трудное. Вдохните, выдохните и начните записывать все, что придает вашему артефакту ценность. Какими качествами обладает идеальный артефакт, нужный вам? Он забавный? Гармоничный? Играбельный? Теперь копните глубже: чем конкретнее то, что вы делаете, тем больше у вас шансов преуспеть в деле. Что означает «забавный»? Для *Super Mario* это имеет одно значение, а для *Civilization* или *Bejeweled* – совсем другое. Я могу назвать больше свойств *хорошего любовного романа эпохи Регентства с элементами мистики,* чем *просто хорошего романа*. Самые простые генераторы – те, в которых можно описать «хорошие» артефакты как набор конкретных свойств.

Теперь ответьте на противоположный вопрос: чем примечателен «плохой» артефакт? Перечислите все, что вы можете придумать, включая относительно неудачные свойства и те, которые портят все безоговорочно. Пометьте звездочкой то, чего *ни в коем случае нельзя допустить*. Это ваши *ограничения*. Наиболее надежными генераторами являются те, в которых можно конкретно описать ограничения.

Теперь у вас есть список желательных свойств и ограничений (того, что вам требуется и что неприемлемо) для ваших артефактов. Нам нужен генератор с *пространством возможностей* (все виды артефактов, которые

он может генерировать), где большинство артефактов наделены хорошими свойствами, а некоторые артефакты (возможно) – плохими. Нам также *понадобится* целый ряд артефактов, а не один и тот же идеальный артефакт, используемый многократно; вопрос о том, насколько *широкий* диапазон вам нужен, вы уже решаете сами как дизайнер (проблема была обозначена Джиллиан Смит (Gillian Smith)[1] и исследуется Майклом Куком (Michael Cook)[2]).

Теперь у нас есть руководство, которое понадобится, чтобы начать строить методы, создающие артефакты.

Альтернативный подход: вышеописанный метод хорош для негибких ситуаций, когда вы заранее знаете, что хотите построить. Во многих ситуациях, таких как игровые джемы[3], прототипы или побочные проекты, вы можете проявлять больше гибкости и импровизировать! Вы вольны начать с метода или некоторых его свободных частей, чтобы выяснить, что они «хотят» генерировать (какие свойства лучше всего для генерации), а затем пересмотреть артефакты, которые вы генерируете, чтобы они в большей мере отвечали тому, что хорошо делает ваш генератор.

Создание вашего «художника в коробке»

Когда разрабатывались редакторы *Spore*, инженеры и дизайнеры тесно сотрудничали с командой художников, чтобы понять, *как* художник или аниматор будет заниматься скульптурированием и текстурированием персонажей. Если бы они могли понять *процесс* и разработать алгоритм, который мог бы следовать этому процессу по требованию, у них был бы «художник в коробке», способный создать процедурно сгенерированных существ, почти столь же изящных, как те, над которыми потрудился аниматор. Дизайнер Хаим Гингольд (Chaim Gingold) объяснил этот процесс в своем выступлении в GDC в 2007 году, и арт-директор Оушен Куигли (Ocean Quigley) использовал аналогичный процесс для эксперимента с видами строительных блоков, которые он хотел бы построить для строительства городов[4].

При создании генератора полезно сесть с людьми, создающими артефакты, которые вам требуются, и попросить, чтобы вас провели через этот процесс. Какие вопросы разработчики задают себе на этом пути? Как они принимают решения? Как объясняют тот или иной выбор? Как описывают различные проблемы, которые надо принимать во внимание? Как называ-

---

1 https://games.soe.ucsc.edu/sites/default/files/smith-expressiverangefdgpcg10.pdf

2 www.gamesbyangelina.org/2016/02/introducing-danesh-part-1/

3 Игровой джем, или game jam, – конкурсы, организуемые игровыми компаниями для независимых команд разработчиков, которые в краткие сроки должны создать игру на конкурс, – и победитель получает не только денежный приз, но и издание собственной инди-игры. – *Прим. перев.*

4 http://oceanquigley.blogspot.com/2009/04/spore-early-rig-block-experiments.html

ют все части того, над чем они работают, и все отношения между частями (их онтологию)?

В некоторых областях науки есть специалисты-практики, которые описывали целые системы, зачастую противоречивые, чтобы наглядно представить то, что они делают. Теория музыки предложила множество систем правил, например для джазовой импровизации, гармоний в стиле Баха или поп-песен. Писатели выработали теории повествования, такие как «путешествие героя», а также основали неформальные ресурсы наподобие TV Tropes. Теория искусства предлагает правило золотого сечения, цветовые гармонии и правила композиции (лично мне эти эстетические стандарты в работе не пригодились, но у вас все может быть по-другому). Ни один из описанных фреймворков не является полноценной системой и не гарантирует создание хороших артефактов, но каждый из них может подарить вам вдохновение и предоставить некое руководство к действию.

Итак, теперь спросите себя: *как с той же задачей справился бы человек?*

От правил к методам генерирования

К сожалению, знать, как человек может выполнить ту или иную задачу, – это не то же самое, что уметь научить этому компьютер. Люди хорошо умеют оценивать, строить догадки и обобщать опыт прошлого. Компьютеры знают только то, что вы им говорите, а многие проблемы требуют гораздо больше неявных знаний, чем кажется; зато компьютеры хорошо выполняют множество вычислений и пробуют много возможностей. Таким образом, методы, которые мы хотим использовать, должны предоставить компьютеру способ решать проблемы примерно так, как это делает человек, или, по крайней мере, путем «отзеркаливания» некоторых человеческих навыков. Методы, подходящие для построения генераторов (методы генерирования), дадут компьютеру некоторые из перечисленных ниже навыков:

- инкапсулировать знания о вариантах (навык A);
- создавать некоторую структуру (навык B);
- кодировать правила с условиями для некоторых опций (A2);
- создавать вариативность в структуре (B2);
- уметь задавать себе вопросы о своих ограничениях («решил ли я эту задачу?») (навык C).

Распределение

Это самый простой вид генеративного метода. У вас есть мешок с некоторым содержимым и область пространства или времени, в которой вы можете его разложить. Методы распределения обычно не имеют большой общей структуры (B), но порой они очень сложны, когда дело касается вы-

бора вариантов распределения (A). Некоторые используют взвешенную случайность, чтобы изменить процент распределения, или «перетасовку колоды» (складывание всех опций в стек и отбрасывание тех, что были использованы), что предотвращает повторный выбор одного и того же варианта. Правила с условиями (A2) также могут быть довольно сложными, но указание произвольных условий трудно реализовать на практике. В большинстве систем имеются тщательно подобранные параметры, которые могут быть установлены для каждого варианта, и условные функции могут просто сравнить фиксированные параметры, чтобы сделать выбор.

Возьмем пример из RPG: блуждающие монстры рассеяны в пространстве (A). Горные обитатели встречаются в соответствующих областях, водные монстры – в воде и т. д. (A2). В их распределении может наблюдаться определенная логика: например, несколько «детских» монстриков ведут к версии «босса». Добыча также распределяется: вы можете с большей вероятностью получить высокоуровневый лут в ситуациях высокого уровня (A2), но все еще остаются некоторые случайные вещи, выбранные из большого списка всевозможных лутов (A).

Метод распределения в музыке и языке не очень хорошо работает. Случайно выбранные строки текста или нотных знаков недостаточно структурированы, чтобы создать смысловую конструкцию. Для сложных по структуре артефактов вместо этого могут потребоваться методы на основе плиток или грамматики, а для артефактов с жестко фиксированной структурой и небольшой вариативностью можно попробовать параметрический подход.

Параметрический метод

У вас уже есть довольно хорошо построенный артефакт, и вы знаете, что его можно слегка улучшить. Скажем, имеется мелодия, и вы можете изменить ее тональность, сделать музыку громче или мягче. Или есть чайник, и вы вольны чуть больше изогнуть носик; корпус можно сделать высоким или коротким, тонким или толстым, а основание – широким или узким. Если у вас имеются инопланетные персонажи, можно сформовать их ноги так, чтобы они были длинными, толстыми, изогнутыми или с плоской стопой; животы могут быть плоскими или вздутыми; голоса также можно менять. Именно так моделируются существа в *No Man's Sky*. Это очень надежная и управляемая технология! 3D-модели часто могут быть кодированы как канал анимации Maya, что позволяет им смешиваться с другими анимациями (трюк *Spore*, используемый в анимации rig-block). Но изменчивость (A) возможна только в заданном диапазоне однонаправленных путей или никакой структурной изменчивости не наблюдается вообще (B2). Вы можете увидеть что-то «новое», но что-либо неожиданное и удивительное – никогда.

Более сложная форма параметрических методов использует другие формы входных данных и может генерировать новые артефакты, основанные не только на числовых, но и на точечных, путевых и графовых входных данных. Когда вы рисуете линию в Photoshop с помощью планшетного пера, ваш путь становится входным для алгоритма, который отображает мазок кисти, учитывая давление, скорость и наклон в качестве параметров в каждой точке. Существа в *Spore* также использовали metaballs, геометрический алгоритм, который может прокладывать гладкие трубы вдоль путей в 3D-пространстве. Другие алгоритмы для заполнения пространства и путей – это диаграмма Вороного (Voronoi patterns), симплекс-шум и шум Перлина (Perlin/Simplex noise), алгоритмы триангуляции, 3D-экструзия или вращение и алгоритм diamond-square для фрактальной местности. Эти алгоритмы особенно хорошо подходят для интерактивных генераторов, поскольку пользователь может предоставить входные параметры для генератора.

Продолжим? На inconvergent.net есть еще больше интригующих образцов, из которых что-то наверняка вас заинтересует, с реализациями с открытым исходным кодом. Однако, хоть эти алгоритмы и поражают воображение, они зачастую слишком малоуправляемы для того, чтобы игрок получил удовольствие от игры и от взаимодействия с довольно плоскими артефактами.

Плиточная основа

Сведем проблему к модульным слотам одинаковых размеров. Есть целый ряд различных решений, которые в данном случае могут быть реализованы вручную и заполнить такие слоты. Здесь создаваемые артефакты – это различные выбранные или упорядоченные наборы предварительно созданных решений. Возможно, вы видели настольную версию игрового поля для таких игр, как *Settlers of Catan* и *Betrayal at the House on The Hill* (или *Civilization*, если приводить цифровой пример). И остров, и особняк строятся каждый раз из одних и тех же плиток, но выкладываются они по-разному, что влияет на ход игры. Я обнаружила один из самых старинных примеров генеративного содержания – это *Musikalisches Würfelspiel*, игра, распространенная в 1750-е годы или даже раньше, с помощью которой пианисты могли собирать «плитки» (в данном случае музыкальные такты), чтобы создавать мелодию вальса[5].

Методы на основе плиток отлично подходят для мелкомасштабной структуры (B), потому что содержимое плитки уже задано, но у них нет гибкости (B2) для мелкомасштабной структуры по той же причине. Крупномасштабная структура труднее поддается контролю: она может быть со-

[5] *Musikalisches Würfelspiel* – система с использованием игральных костей для случайного «генерирования» музыки из заранее составленных вариантов. Подобные игры были довольно популярны в Западной Европе в XVIII в. – *Прим. перев.*

вершенно случайной. Вы можете задать довольно подробные ограничения в том, что касается сочетаемости плиток, но тогда вам может понадобиться сложный алгоритм для решения проблемы совместимости («пляжная плитка может стоять рядом с плиткой джунглей, но должна находиться на расстоянии не менее двух плиток от плитки с рекой»). Отдельные плитки имеют очень жесткую инкапсуляцию возможных вариантов (A), потому что каждая разновидность плиток должна быть создана человеком. Эти системы не имеют достаточно информации для того, чтобы разработать принципиально новые плитки. Плиточная основа удобна для решения задач, которые могут быть разбиты на небольшие фрагменты, где важна внутренняя структура и при этом могут создаваться интересные (без нарушения ограничений) комбинации при объединении в различном порядке.

Вас интересуют дополнительные материалы по генерации аналогового контента? Описание контента настольных игр и ролевых игр вы найдете в «Аналоговой истории процедурного поколения» Джиллиан Смит[6] (Gillian Smith. An Analog History of Procedural Generation) и исследовании комиксов Криса Мартенса (Chris Martens)[7] (см. рис. 1.1).

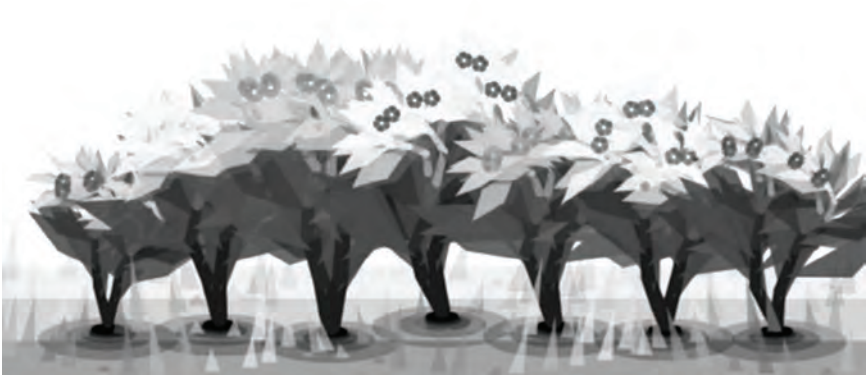

Рис. 1.1

Грамматики

Грамматики – один из моих самых любимых генеративных методов: они позволяют создавать очень глубокие и сложные структуры, и при этом я могу хорошо контролировать имеющиеся у меня варианты. Грамматики – это способ показать, что большие сложные объекты состоят из других объектов, а те могут быть скомпонованы на основе еще более мелких и простых. *Nested* от Ортейла – прекрасный пример[8]. Вселенная – это множество галактик, включающих в себя планеты, состоящие из континентов,

[6] http://sokath.com/main/files/1/smith-fdg15.pdf

[7] http://lambdamaphone.blogspot.com/2015/12/generativity-interpretation-study-of.html

[8] Ортейл (Orteil) – французский веб- и javascript-разработчик. Он создал исключительно популярную браузерную игру *Cookie Clicker*, а также вышеупомянутую *Nested* и *Turtle Toy*. Его творческие эксперименты доступны на сайте http://orteil.dashnet.org. – *Прим. перев.*

наполненных людьми, которые сотканы из атомов, а также не лишены мыслей, снов, воспоминаний… Каждый символ (т. е. каждый тип объекта, как показано на рис. 1.1) имеет распределение подсимволов, из которых он может быть составлен. Когда он «распаковывается», он должен выбрать один из этих вариантов (и любые подсимволы), а затем распаковать рекурсивно. Грамматики позволяют легко кодировать знания об определенном артефакте, его структуре и вариантах в одном фрагменте данных. Мне они так нравятся, что я подготовила специальную библиотеку *Tracery*, чтобы облегчить людям работу с ними. Она использовалась для создания сервиса хостинга *Twitterbot* и множества больших и странных креативных твиттер-ботов, в том числе нескольких моих (см. также главу 27 этой книги). Недостаток грамматик в том, что они не имеют способа обработки ограничений, если только ограничения не закодированы в самих грамматиках (скажем, если кровать может быть только в доме, значит, только дом может иметь кровать в качестве дочернего элемента). Грамматикам сложнее кодировать высокоуровневые отношения между различными объектами, порожденными в разных точках грамматики. Если вы хотите, чтобы предзнаменование смерти злодея содержалось в самом начале грамматики, решить такую задачу будет непросто, и вы можете использовать для этого «большой молот» решателя ограничений.

Решатели ограничений

Решатели ограничений – это очень мощные и достаточно новые инструменты в нашем арсенале. Их удобно использовать, когда у вас много жестких ограничений и много гибких и сложных структур, но вы не знаете, как построить структуры таким образом, чтобы обязательно разрешать все ваши ограничения.

Самая старая и простая тактика – метод грубой силы: подготовьте все возможные варианты контента, переключите каждый переключатель, создайте альтернативную вселенную, в которой вы приняли каждое новое решение, и проверяйте свои ограничения, пока не найдете то, которое работает. Некоторые проблемы удастся решить таким образом, но, как скажет вам любой математик, при обилии вариантов число возможных артефактов превысит число атомов во Вселенной, и поиск будет очень медленным.

Зачастую существуют кратчайшие пути, которые вы можете использовать в зависимости от того, как структурированы ваши ограничения (мне не нужно выбирать вкус мороженого в те дни, когда я не выхожу за мороженым). Но автору требуется чересчур много времени, чтобы прописать это вручную (не верите – спросите разработчика *Storyteller*). К счастью, многие математики и логики нашли забавным сам процесс решения проблемы и придумали решатели общего назначения. Подключите свои ограничения, структуры и параметры (на языке, который решатели могут понять), и они

будут искать все возможные кратчайшие пути, чтобы сократить область применения грубой силы и обеспечить медленное, но выполнимое решение за время игры.

Поскольку эти инструменты большие, громоздкие и новые, они все еще с трудом подключаются во многих игровых движках, и учебных материалов еще не так много. Адам Смит (Adam Smith) проводит хорошую образовательную пропаганду для метода «программирование наборов ответов» (Answer Set Solving), что является особенно мощным методом. *Craft* Йана Хорсвилла (Ian Horswill) – это генератор ограниченных случайных чисел с некоторой поддержкой, который недавно был перенесен на Javascript. Вот увидите, эти редкие, но мощные инструменты в будущем получат широкое распространение!

Агенты и моделирование

А вот тут начинается самое странное. Помните, я сказала, что мы могли бы посмотреть, как люди решают задачи, чтобы обеспечить генераторам вдохновение? Угадайте, в чем проблема. Люди не единственные, кто решает задачи! Некоторые алгоритмы решают задачи, основанные на колониальном поведении муравьев или социальных коммуникациях людей. Для многих агентов и симуляторов источником вдохновения становится природа в самых разных ее проявлениях: стаи птиц, эволюция, бактерии, нейроны, города. Вот лишь некоторые из моих любимых примеров.

Имитация естественного движения (steering behaviors) может создавать удивительно сложные движения толпы. «Тележки Брайтенберга» (Braitenberg vehicles) – это в своей первооснове мысленный эксперимент: простые машины с двумя фоточувствительными глазами и двумя колесами «управляют» собой, просто активируя одно колесо больше, чем другое, так что сила асимметрии изменяет их направление. Хоть и неодушевленные, они могут показать эмоцию «страх» и статус «приключение»; были портированы на многие языки и даже на физических роботов.

В алгоритме *Boids* имитация естественного движения тележек Брайтенберга применяется к стаям птиц и рыб (и гну в анимационном фильме «Король Лев»)[9]. Каждая птица помогает поддерживать форму стаи путем высчитывания и прикладывания своих собственных усилий для сцепления, выравнивания и разъединения. Небольшие вариации значений в настройке каждой птицы могут генерировать новое поведение целой стаи.

Я также использовала управляющие силы для создания процедурного танца (рис. 1.2): здесь вместо силы поддержания формы птичьей стаи действуют ритмические силы – в такт музыке. Управляющие силы могут сделать намного больше, чем просто найти и проложить себе путь, и, надо думать, их возможности пока не исследованы в полной мере.

[9] *Boids* – компьютерная программа, разработанная Крейгом Рейнольдсом (Craig Reynolds) в 1986 году для имитации движения птиц. – *Прим. перев.*

Рис. 1.2

Генетические алгоритмы не генерируют контент, так как вам все еще нужен генератор, но они направляют этот генератор к более ограниченному заполнению и желательному содержанию, производящему свойства. Я использовала этот метод в приложении для эволюции растений (рис. 1.3, 1.4). Генетическому алгоритму нужно следующее:

1) что-то, что вы можете изменить (генотип);
2) что-то, о чем вы можете судить (фенотип);
3) способ превратить первое во второе.

Рис. 1.3

Для цветов в моем приложении генотип представляет собой массив дробных чисел. Этот массив подается в параметрический генератор для создания достаточно неплохо анимированного цветка (превращение генотипа в фенотип). Пользователи определяют, какие цветы им нравятся (что-то, о чем вы можете судить). Они выбирают самый привлекательный цветок; его исходный генотип клонируется и мутирует (что-то, что вы можете изменить), и следующее поколение рождается от мутировавших экземпляров; со временем происходит эволюция. Половое размножение широко распространено в генетических алгоритмах, но в них присутствуют и многие другие интересные виды размножения и метаэвристики. Существует масса актуальных исследований на эту тему.

Рис. 1.4

Клеточные автоматы полагаются на множество простых агентов, работающих параллельно. Классический пример – игра *Conway's Game of Life*, в которой многие крошечные автоматы в сетке, каждый с очень простым поведением, могут вызвать так много странных явлений, что их обнаружение и каталогизация для многих математиков превратились в полноценное хобби. Этот метод может привести, скажем, к таким результатам, как на рис. 1.5. Клеточные автоматы с более сложными правилами используются для создания *Dwarf Fortress*, классической игры *Powder*, и с появлением симуляторов, использующих вокселы, они начинают новую жизнь в качестве основного двигателя *Minecraft*[10].

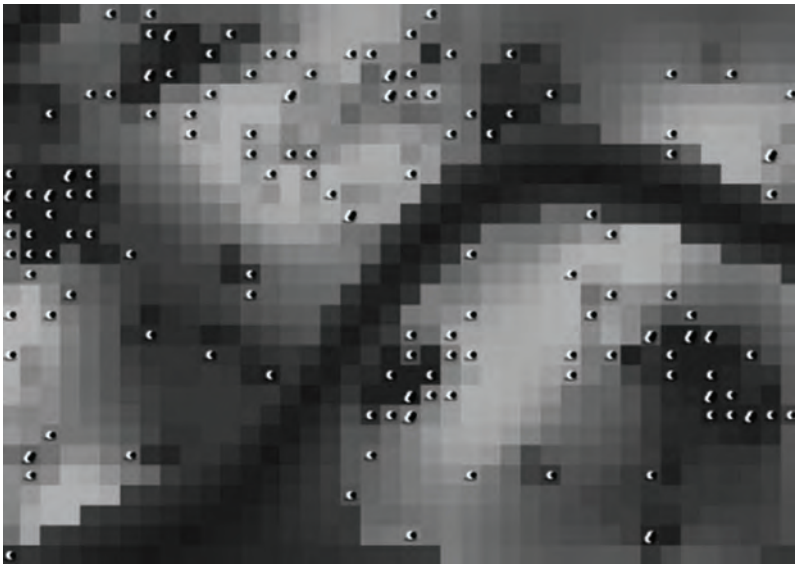

Рис. 1.5

[10] Воксел (voxel) – элемент объемного изображения, является аналогом двумерных пикселей для трехмерного пространства. По сути, это просто один из 3D-кирпичиков, из которых можно собирать что-то новое и большее. На этом и основан гейм-плей *Minecraft*. – *Прим. перев.*

И вот вы приступили к генерированию. Вы просмотрели список генеративных методов, у вас теперь есть свой список ограничений и свойств, и вы построили свой генератор! Что же дальше?

Как могут сбоить генераторы

Что-то явно пошло не по плану. Контент выглядит отвратительно. Все какое-то одинаковое. И напоминает гениталии. Полный провал! Некоторые проблемы решить проще, некоторые – сложнее. Вот ряд сложных проблем, с которыми вы столкнетесь.

Первая: вы можете путем вычислений определить, что что-то идет не так. Решение заключается в создании нового контента до тех пор, пока это ограничение не будет нарушаться. Возможно, вы хотите, чтобы центр тяжести сгенерированного стула находился над опорными точками (например, ножками), чтобы стул не падал. Это можно рассчитать с помощью физики моделирования, поэтому, если сгенерированный стул не удался, создайте новый, пока вам не улыбнется удача. Этот подход называется «генерация и тестирование».

Что может пойти не так с генерацией и тестированием? А вдруг ни один стул пройдет этот тест? Возможно, содержание, которое проходит тест, очень редко генерируется или существует слишком много ограничений (у вас есть ограничения по толщине материала, по стоимости, симметрии, и комфорту и так далее). Каждый стул может удовлетворять большинству ограничений, но при значительном их количестве большинство стульев все равно не будет удовлетворять одному-двум ограничениям. Может быть, вам нужен решатель ограничений. Или вам нужно ограничить свой генератор, чтобы он был более консервативен в выборе, хотя таким образом вы несколько сужаете пространство возможностей.

Вторая проблема: вы не можете адекватно описать ограничения. Это чрезвычайно распространенная ситуация, потому что есть масса ненужных нам вещей, но мы не можем написать правила в духе «когда увижу – пойму, нужно или нет». Такие соображения нельзя использовать в качестве серьезного аргумента.

Такое-то определение в характеристике персонажа звучит оскорбительно? Не слишком ли эта палитра напоминает цвета фирменного логотипа? Это изображение вызывает неприличные ассоциации? Это просто выглядит уродливо? Боюсь, подобные проблемы почти нерешаемы. Если вы не можете определить «плохой» контент, его невозможно отфильтровать, особенно когда пользователи пытаются обойти алгоритм обнаружения. В этом случае лучший путь – построить генератор, который максимально снизит вероятность генерации нежелательного контента. Это тоже ограничивает ваше пространство возможностей: например, приходится поступиться словами, которые сами по себе нейтральны, но в сочетании друг с другом образуют бранное выражение.

Эстетика: самая сложная задача

Наиболее распространенный вариант неудачной работы генераторов – когда они производят контент, который не представляет интереса. Что такое «интересный контент»? Это зависит от ситуации. Очень немногие генераторы производят только что-то одно. Большинство из них генерируют кратные числа, но твиттер-бот, размещающий публикации каждый час, будет генерировать больше контента, чем генератор текста, выводящий по одному роману в каждый NaNoGenMo[11]. Таким образом, достижение новизны с помощью первого твиттер-бота будет более сложной задачей, потому что производится так много артефактов, что любой из них, вероятно, начнет казаться менее особенным.

Ваш алгоритм может генерировать 18 446 744 073 709 551 616 планет. У каждой свои нюансы, но, поскольку игрок быстро изучает их, будет ли слишком заметна разница? Я это называю проблемой 10 000 чашек овсянки. Легко можно создать 10 000 тарелок овсянки, где овсянка выложена по-разному; в математическом понимании все эти тарелки уникальны. Но пользователь, скорее всего, просто увидит много овсянки. Перцептивная уникальность, обеспечивающая разность восприятия, – реальная метрика, и достичь этого чертовски трудно.

В некоторых ситуациях достаточно просто перцептивной дифференциации, и ее куда легче добиться. Перцептивная дифференциация – это ощущение того, что данный фрагмент содержания не идентичен предыдущему. Пользователь, взглянув на аллею, может сразу заметить, что деревья почти одинаковые или так мало отличаются друг от друга, что выглядят неестественно. Относительное разнообразие несет в себе эстетическую ценность, даже если ни одно дерево не является особенно запоминающимся.

Перцептивная уникальность гораздо сложнее. Это различие между, скажем, артистом массовки, и характерным актером. Должен ли каждый артефакт иметь свою индивидуальность? Это потребовало бы чересчур большой работы, да и у пользователей глаза будут разбегаться. Не каждому суждено блистать. Многие артефакты должны всего лишь служить фоном для других, более приметных и ярких.

Характерные артефакты – это уже предмет отдельного обсуждения, но определенные эстетические принципы в любом случае помогут вам создавать объекты с хорошо угадываемыми смыслами. Людям нравится детализация: взрыхленная почва у основания дерева или трава, растущая около надгробного камня, придают натуралистичность пейзажу. Эти ню-

[11] National Novel Generation Month – Национальный месяц генерации романа, проект Дариуса Каземи. Он предложил в Твиттере потратить месяц ноябрь на написание кода, который генерирует роман из более чем 50 тыс. слов, а затем выложить код и роман на всеобщее обозрение. Эта идея была воспринята с таким энтузиазмом, что теперь NaNoGenMo проводится каждый год. Можно найти примеры кода участников на GitHub, там же приводится хронология проекта. – *Прим. перев.*

ансы подсказывают нам, что за артефактом скрывается живой мир. В популярном исследовании Кевина Линча (Kevin Lynch) «Образ города» (Image of the City) показано, что города запоминаются и становятся знаковыми благодаря определенным деталям. Возможно, есть и другие эстетические правила, которые вы сами для себя откроете.